Monday Sep. 24

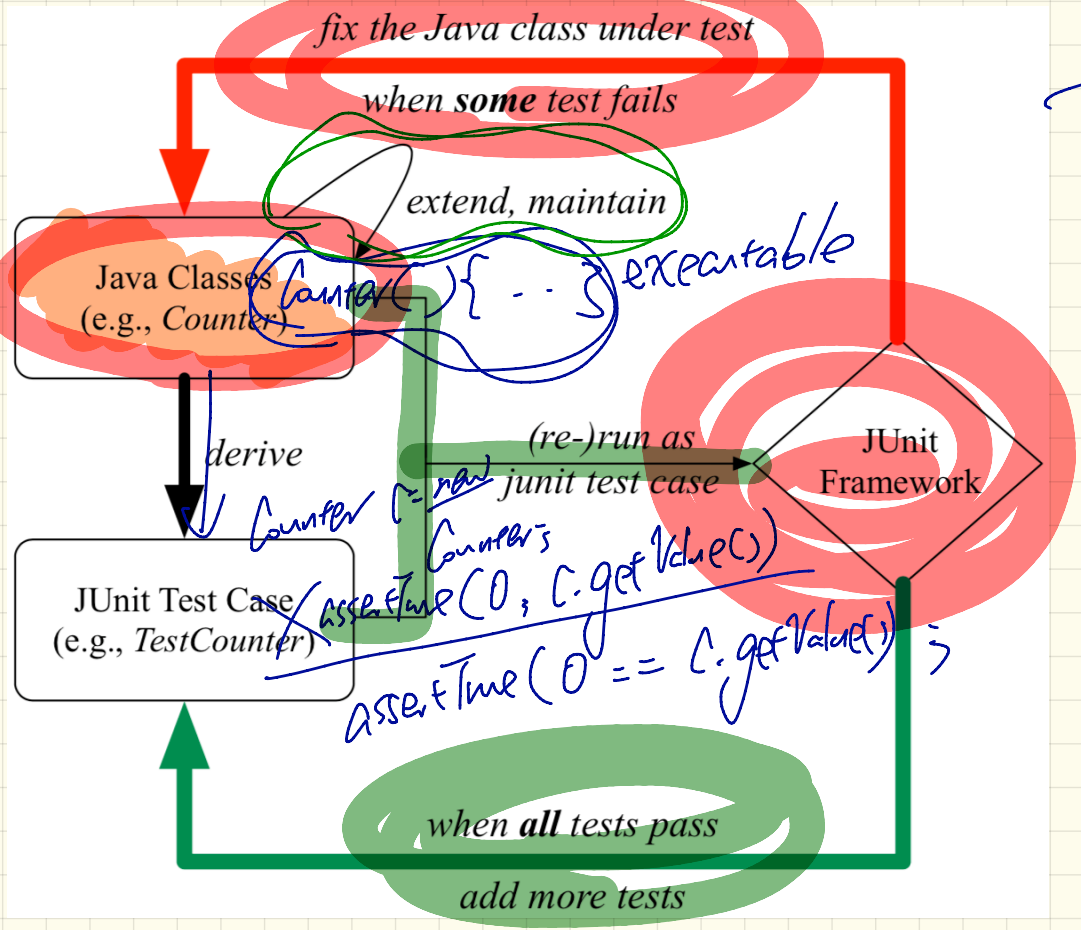
Lecture 6

- Mandatory Lab Session today
  (Submission within 20 minutes)
- Lab Test I Guide
  - ~ Birthday Book
  - ~ Encapsulation
  - ~ Expectation & Strategy
  - ~ equals method

JUnit Test Case 4

```java
@Test
public void testIncDecFromMiddleValues() {
    Counter c = new Counter();
    try {
        for(int i = Counter.MIN_VALUE; i < Counter.MAX_VALUE; i ++) {
            int currentValue = c.getValue();
            c.increment();
            assertEquals(currentValue + 1, c.getValue());
        }
        for(int i = Counter.MAX_VALUE; i > Counter.MIN_VALUE; i --) {
            int currentValue = c.getValue();
            c.decrement();
            assertEquals(currentValue - 1, c.getValue());
        }
    }
    catch(ValueTooLargeException e) {
        fail("ValueTooLargeException is thrown unexpectedly");
    }
    catch(ValueTooSmallException e) {
        fail("ValueTooSmallException is thrown unexpectedly");
    }
}
```

Test-Driven Development (TDD)



PointV1

PointV2

```
int i = 1;
int j = 2;
assertTrue( i == j );
```

X

```
assertEquals( i, j );
```

```
Person p1, p2;
assertEquals(p1, p2);
```

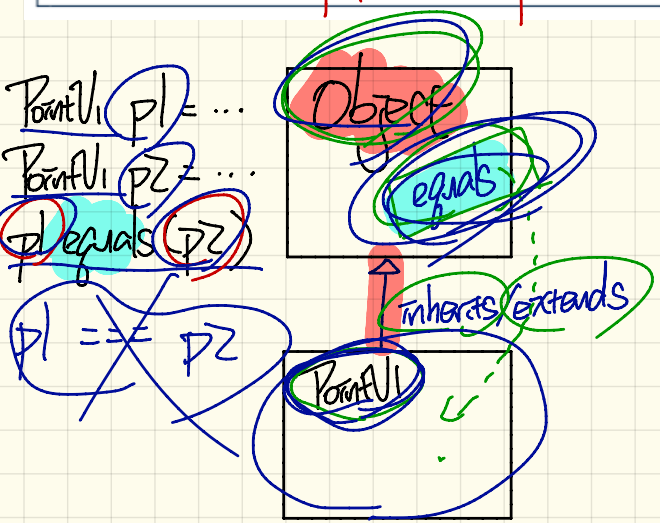
① `p1 == p2;`

② `p1.equals(p2)`

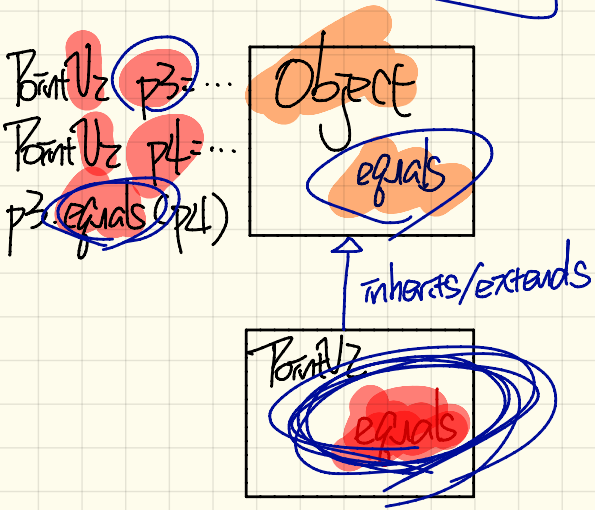
equals method in Object class

Case 1: equals not overridden

```
boolean equals(Object other) {
  return (this == other);
}
```



Case 2: equals overridden



equals method case 1: calling default version

from Object class

```
boolean equals(Object other) {
  return (this == other);
}
```

```
class PointV1 {
  double x; double y;
  PointV1(double x, double y) { this.x = x; this.y = y; }
}
```

```
PointV1 p1 = new PointV1(2, 3);
PointV1 p2 = new PointV1(2, 3);
System.out.println( p1 == p2 ); /* false */
System.out.println( p1.equals(p2) ); /* false */
```

equals method case 2: overridding default version

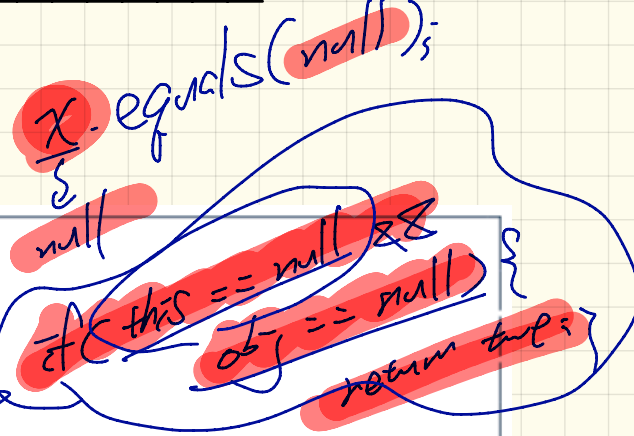
Step 1: x.equals(x) == True

```
class PointV2 {
  double x; double y;
  public boolean equals (Object obj) {
    if(this == obj) { return true; }
    if(obj == null) { return false; }
    if(this.getClass() != obj.getClass()) { return false; }
    Point other = (PointV2) obj;
    return this.x == other.x && this.y == other.y; } }
```

```
String s = "(2, 3)";
PointV2 p1 = new PointV2(2, 3); PointV2 p2 = new PointV2(2, 3);
System.out.println(p1.equals(p1)); /* true */
System.out.println(p1.equals(null)); /* false */
System.out.println(p1.equals(s)); /* false */
System.out.println(p1 == p2); /* false */
System.out.println(p1.equals(p2)); /* true */
```

equals method case 2: overridding default version

Step 2: x.equals(null) == False

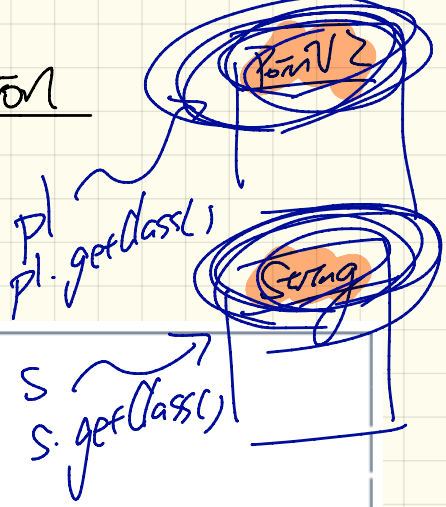



```
class PointV2 {
  double x; double y;
  public boolean equals (Object obj) {
    if(this == obj) { return true; }
    if(obj == null) { return false; }
    if(this.getClass() != obj.getClass()) { return false; }
    Point other = (PointV2) obj;
    return this.x == other.x && this.y == other.y; } }
```

```
String s = "(2, 3)";
PointV2 p1 = new PointV2(2, 3); PointV2 p2 = new PointV2(2, 3);
System.out.println(p1.equals(p1));   /* true */
System.out.println(p1.equals(null)); /* false */
System.out.println(p1.equals(s));    /* false */
System.out.println(p1 == p2);        /* false */
System.out.println(p1.equals(p2));   /* true */
```

equals method case 2: overridding default version

Step 3: apple.equals(banana) == false




```
class PointV2 {
  double x; double y;
  public boolean equals (Object obj) {
    if(this == obj) { return true; }
    if(obj == null) { return false; }
    if(this.getClass() != obj.getClass()) { return false; }
    Point other = (PointV2) obj;
    return this.x == other.x && this.y == other.y; } }
```

dynamic type

```
String s = "(2, 3)";
PointV2 p1 = new PointV2(2, 3); PointV2 p2 = new PointV2(2, 3);
System.out.println(p1.equals(p1));   /* true */
System.out.println(p1.equals(null)); /* false */
System.out.println(p1.equals(s));    /* false */
System.out.println(p1 == p2);        /* false */
System.out.println(p1.equals(p2));   /* true */
```

p1 == s

X

equals method case 2: overridding default version

Step 4: apple.equals(apple) depends on your def.

```
class PointV2 {
  double x; double y;
  public boolean equals(Object obj) {
    if(this == obj) { return true; }
    if(obj == null) { return false; }
    if(this.getClass() != obj.getClass()) { return false; }
    Point other = (PointV2) obj;
    return this.x == other.x && this.y == other.y; } }
```

declaration

static type

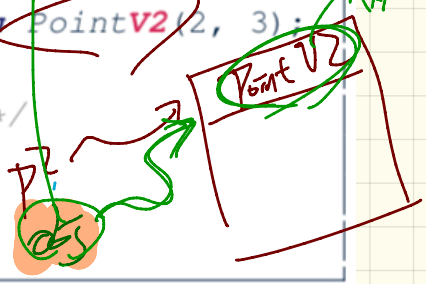
Static Type → Object

Dynamic Type → PointV2

```
String s = "(2, 3)";
PointV2 p1 = new PointV2(2, 3); PointV2 p2 = new PointV2(2, 3);
System.out.println(p1.equals(p1));   /* true */
System.out.println(p1.equals(null)); /* false */
System.out.println(p1.equals(s));    /* false */
System.out.println(p1 == p2);        /* false */
System.out.println(p1.equals(p2));   /* true */
```

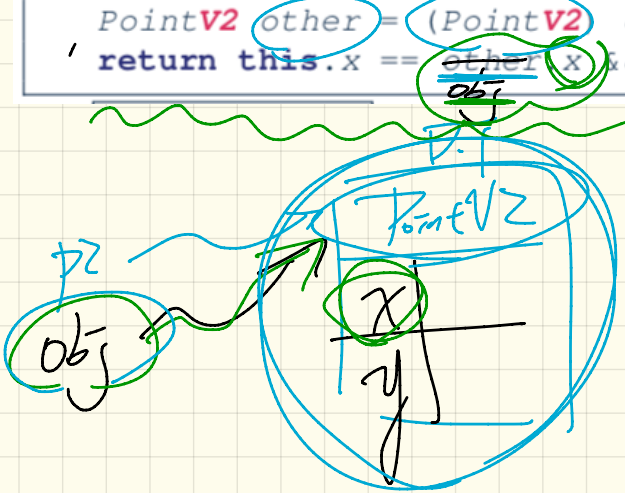
S.T.

D.T. Dynamic



Type Casting in Step 4 of Case 2

```
class PointV2 {
  boolean equals(Object obj) { ...
    if(this.getClass() != obj.getClass()) { return false; }
    PointV2 other = (PointV2) obj;
    return this.x == other.x && this.y == other.y; } }
```



will not compile

obj has ST Object which means it cannot call att/met defined in its DT (PointV2)

Equality on Person

```
class Person {
  String firstName; String lastName; double weight; double height;
  boolean equals(Object obj) {
    if(this == obj) { return true; }
    if(obj == null || this.getClass() != obj.getClass()) {
      return false; }
    Person other = (Person) obj;
    return
      this.weight == other.weight && this.height == other.height
      && this.firstName.equals(other.firstName)
      && this.lastName.equals(other.lastName); } }
```

for String -

Equality on PersonCollector

```
class PersonCollector {
  Person[] persons; int nop; /* number of persons */
  public PersonCollector() { ... }
  public void addPerson(Person p) { ... }
}
```

Redefine/Override the `equals` method in `PersonCollector`.

```
boolean equals(Object obj) {
  if(this == obj) { return true; }
  if(obj == null || this.getClass() != obj.getClass()) {
    return false; }
  PersonCollector other = (PersonCollector) obj;
  boolean equal = false;
  if(this.nop == other.nop) {
    for(int i = 0; equal && i < this.nop; i ++) {
      equal = this.persons[i].equals(other.persons[i]); } }
  return equal;
}
```